

# Robust Multi-Robot Optimal Path Planning with Temporal Logic Constraints

Alphan Ulusoy<sup>\*</sup>   Stephen L. Smith<sup>†</sup>   Xu Chu Ding<sup>‡</sup>   Calin Belta<sup>\*</sup>

**Abstract**—In this paper we present a method for automatically planning robust optimal paths for a group of robots that satisfy a common high level mission specification. Each robot's motion in the environment is modeled as a weighted transition system, and the mission is given as a Linear Temporal Logic (LTL) formula over a set of propositions satisfied by the regions of the environment. In addition, an optimizing proposition must repeatedly be satisfied. The goal is to minimize the maximum time between satisfying instances of the optimizing proposition while ensuring that the LTL formula is satisfied even with uncertainty in the robots' traveling times. We characterize a class of LTL formulas that are robust to robot timing errors, for which we generate optimal paths if no timing errors are present, and we present bounds on the deviation from the optimal values in the presence of errors. We implement and experimentally evaluate our method considering a persistent monitoring task in a road network environment.

## I. INTRODUCTION

The classic motion planning problem considers missions where a robot must reach a goal state from an initial state while avoiding obstacles. Temporal logics, on the other hand, provide a powerful high-level language for specifying complex missions for groups of robots [1], [2], [3], [4], [5]. Their power lies in the wealth of tools from model checking [6], [7], which can be leveraged to generate robot paths satisfying desired mission specifications. Alternatively, if the mission cannot be satisfied, the tools can be used produce a certificate, or counter-example, which proves that the mission is not possible. However, in robotics the goal is typically to plan paths that not only complete a desired mission, but which do so in an optimal manner. In our earlier work [8] we considered Linear Temporal Logic (LTL) specifications, and a particular form of cost function, and provided a method for computing optimal robot paths for a single robot. We then extended this approach to multi-robot problems by utilizing timed automata [9].

The main difficulty in moving from a single robot to multiple robots is in synchronizing the motion of the robots, or in allowing the robots to move asynchronously. In [10], the authors propose a method for decentralized motion of multiple robots by restricting the robots to take transitions (*i.e.*, travel along edges in the graph) synchronously.

Once every robot has completed a transition, the robots can synchronously make the next transition. While such an approach is effective for satisfying the LTL formula, it does not lend itself to optimizing the robot motion, since robots must spend extra time for synchronization. In [9] we approached this problem by describing the motion of the group of robots in the environment as a timed automaton. This description allowed us to represent the relative position between robots. Such information is necessary for optimizing the robot motion. After providing a bisimulation [11] of the infinite-dimensional timed automaton to a finite dimensional transition system we were able to apply our results from [8] to compute an optimal run.

However, enabling the asynchronous motion of robots introduces issues in the robustness, and thus implementability of the multi-robot paths. Timed-automata rely heavily on the assumption that the clocks (or for robots, the speeds), are known exactly. If the clocks drift by even an infinitesimally small amount, then the reachability analysis developed for timed-automata is no longer correct [12], [13]. The intuition behind this is that if the robot speeds are not exactly equal to those used for planning, then two robots can complete tasks in a different order than was specified in the plan. This switch in the order of events may result in the violation of the global mission specification.

In this paper, we address this issue by characterizing a class of LTL formulas that are robust to such timing errors. For simplicity of presentation, we assume that each robot moves among the vertices of an environment modeled as a graph. However, by using feedback controllers for facet reachability in polytopes [14] the method developed in this paper can be extended to robots with continuous dynamics traversing an environment with polytopic partitions. The characterization relies on the concept of trace-closedness of languages, which was first applied in multi-robot planning in [15]. For these languages, we can guarantee that any deviation from the planned order of events due to uncertainties in the speeds of robots will not result in the violation of the global specification.

The contribution of this paper is to present a method for generating paths for a group of robots satisfying general LTL formulas, which are robust to uncertainties in the speeds of robots, and which perform within a known bound of the optimal value. We focus on minimizing a cost function that captures the maximum time between satisfying instances of an *optimizing proposition*. The cost is motivated by problems in persistent monitoring and in pickup and delivery problems. Our solution relies on using the concept of trace-closedness

This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, NSF CNS-0834260, NSF CNS-1035588 and NSERC.

<sup>\*</sup> Hybrid and Networked Systems Laboratory, Boston University, Boston, MA 02215 (alphan@bu.edu, cbelta@bu.edu)

<sup>†</sup> Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (stephen.smith@uwaterloo.ca)

<sup>‡</sup> Embedded Systems and Networks, United Technologies Research Center, East Hartford, CT 06108 (dingx@utrc.utc.com)

to characterize the class of LTL formulas for which a robust solution exists. For formulas in this class, we utilize a similar method as in [9] to generate robot plans. We then propose periodic synchronization of the robots to optimize the cost function in the presence of timing errors. We provide results from an implementation on a robotic test-bed, which shows the utility of the approach in practice.

The organization of the paper is as follows. In Section II, we give some preliminaries in formal methods and trace-closed languages. In Section III, we formally state the motion planning problem for a team of robots, and we present our solution in Section IV. In Section V, we present a hardware implementation for a team of robots performing persistent data gathering missions in a road network environment. Finally, in Section VI, we conclude with final remarks.

## II. PRELIMINARIES

For a set  $\Sigma$ , we use  $|\Sigma|$ ,  $2^\Sigma$ ,  $\Sigma^*$ , and  $\Sigma^\omega$  to denote its cardinality, power set, set of finite words, and set of infinite words, respectively. Moreover, we define  $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$  and denote the empty string by  $\emptyset$ .

**Definition II.1 (Transition System).** A (weighted) transition system (TS) is a tuple  $\mathbf{T} := (\mathcal{Q}_T, q_T^0, \delta_T, \Pi_T, \mathcal{L}_T, w_T)$ , where

- (i)  $\mathcal{Q}_T$  is a finite set of states;
- (ii)  $q_T^0 \in \mathcal{Q}_T$  is the initial state;
- (iii)  $\delta_T \subseteq \mathcal{Q}_T \times \mathcal{Q}_T$  is the transition relation;
- (iv)  $\Pi_T$  is a finite set of atomic propositions (observations);
- (v)  $\mathcal{L}_T : \mathcal{Q}_T \rightarrow 2^{\Pi_T}$  is a map giving the set of atomic propositions satisfied in a state;
- (vi)  $w_T : \delta_T \rightarrow \mathbb{R}_{>0}$  is a map that assigns a positive weight to each transition.

We define a run of  $\mathbf{T}$  as an infinite sequence of states  $r_T = q^0 q^1 \dots$  such that  $q^0 = q_T^0$ ,  $q^k \in \mathcal{Q}_T$  and  $(q^k, q^{k+1}) \in \delta_T$  for all  $k \geq 0$ . A run generates an infinite word  $\omega_T = \mathcal{L}(q^0)\mathcal{L}(q^1)\dots$  where  $\mathcal{L}(q^k)$  is the set of atomic propositions satisfied at state  $q^k$ .

**Definition II.2 (LTL Formula).** An LTL formula  $\phi$  over the atomic propositions  $\Pi$  is defined inductively as follows:

$$\phi ::= \top \mid \alpha \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \mid \mathbf{X} \phi \mid \phi \mathbf{U} \phi$$

where  $\top$  is a predicate true in each state of a system,  $\alpha \in \Pi$  is an atomic proposition,  $\neg$  (negation),  $\vee$  (disjunction) and  $\wedge$  (conjunction) are standard Boolean connectives, and  $\mathbf{X}$  and  $\mathbf{U}$  are temporal operators.

LTL formulas are interpreted over infinite words (generated by the transition system  $\mathbf{T}$  from Def. II.1). Informally,  $\mathbf{X} \alpha$  states that at the next position of a word, proposition  $\alpha$  is true. The formula  $\alpha_1 \mathbf{U} \alpha_2$  states that there is a future position of the word when proposition  $\alpha_2$  is true, and proposition  $\alpha_1$  is true at least until  $\alpha_2$  is true. From these temporal operators we can construct two other temporal operators: Eventually (i.e., future),  $\mathbf{F}$  defined as  $\mathbf{F} \phi := \top \mathbf{U} \phi$ , and Always (i.e., globally),  $\mathbf{G}$ , defined as  $\mathbf{G} \phi := \neg \mathbf{F} \neg \phi$ . The formula  $\mathbf{G} \phi$  states that  $\phi$  is true at all positions of the word; the formula

$\mathbf{F} \phi$  states that  $\phi$  eventually becomes true in the word. More expressivity can be achieved by combining the temporal and Boolean operators. We say a run  $r_T$  satisfies  $\phi$  if and only if the word generated by  $r_T$  satisfies  $\phi$ .

**Definition II.3 (Büchi Automaton).** A Büchi automaton is a tuple  $\mathbf{B} := (\mathcal{S}_B, \mathcal{S}_B^0, \Sigma_B, \delta_B, \mathcal{F}_B)$ , consisting of

- (i) a finite set of states  $\mathcal{S}_B$ ;
- (ii) a set of initial states  $\mathcal{S}_B^0 \subseteq \mathcal{S}_B$ ;
- (iii) an input alphabet  $\Sigma_B$ ;
- (iv) a non-deterministic transition relation  $\delta_B \subseteq \mathcal{S}_B \times \Sigma_B \times \mathcal{S}_B$ ;
- (v) a set of accepting (final) states  $\mathcal{F}_B \subseteq \mathcal{S}_B$ .

A run of  $\mathbf{B}$  over an input word  $\omega = \omega^0 \omega^1 \dots$  is a sequence  $r_B = s^0 s^1 \dots$ , such that  $s^0 \in \mathcal{S}_B^0$ , and  $(s^k, \omega^k, s^{k+1}) \in \delta_B$ , for all  $k \geq 0$ . A Büchi automaton  $\mathbf{B}$  accepts a word over  $\Sigma_B$  if and only if at least one of the corresponding runs intersects with  $\mathcal{F}_B$  infinitely many times. For any LTL formula  $\phi$  over a set  $\Pi$ , one can construct a Büchi automaton with input alphabet  $\Sigma_B = 2^\Pi$  accepting all and only words over  $2^\Pi$  that satisfy  $\phi$ .

**Definition II.4 (Prefix-Suffix Structure).** A prefix of a run is a finite path from an initial state to a state  $q$ . A periodic suffix is an infinite run originating at the state  $q$  reached by the prefix, and periodically repeating a finite path, which we call the suffix cycle, originating and ending at  $q$ , and containing no other occurrence of  $q$ . A run is in prefix-suffix form if it consists of a prefix followed by a periodic suffix.

**Definition II.5 (Language).** The set of all the words accepted by an automaton  $\mathbf{B}$  is called the language recognized by the automaton and is denoted by  $L_B$ .

**Definition II.6 (Distribution).** Given a set  $\Sigma$ , the collection of subsets  $\Sigma_i \subseteq \Sigma$ ,  $\forall i = 1, \dots, m$  is called a distribution of  $\Sigma$  if  $\cup_{i=1}^m \Sigma_i = \Sigma$ .

**Definition II.7 (Projection).** For a word  $\omega \in \Sigma^\infty$  and a subset  $\Sigma_i \subseteq \Sigma$ ,  $\omega \upharpoonright_{\Sigma_i}$  denotes the projection of  $\omega$  onto  $\Sigma_i$ , which is obtained by removing all the symbols in  $\omega$  that are not in  $\Sigma_i$ . For a language  $L \subseteq \Sigma^\infty$  and a subset  $\Sigma_i \subseteq \Sigma$ ,  $L \upharpoonright_{\Sigma_i}$  denotes the projection of  $L$  onto  $\Sigma_i$ , which is the set of projections of all words in  $L$  onto  $\Sigma_i$ , i.e.,  $\{\omega \upharpoonright_{\Sigma_i} \mid \omega \in L\}$ .

**Definition II.8 (Trace-Closed Language).** Given the distribution  $\{\Sigma_1, \dots, \Sigma_m\}$  of  $\Sigma$  and the words  $\omega, \omega' \in \Sigma^\infty$ ,  $\omega'$  is trace-equivalent to  $\omega$ , denoted  $\omega' \sim \omega$ , iff their projections onto each one of the subsets in the given distribution are equal, i.e.,  $\omega \upharpoonright_{\Sigma_i} = \omega' \upharpoonright_{\Sigma_i}$  for each  $i = 1, \dots, m$ . For  $\{\Sigma_1, \dots, \Sigma_m\}$ , the trace-equivalence class of  $\omega$  is given by  $[\omega] = \{\omega' \in \Sigma^\infty \mid \omega' \upharpoonright_{\Sigma_i} = \omega \upharpoonright_{\Sigma_i} \forall i = 1, \dots, m\}$ . Finally, a trace-closed language over  $\{\Sigma_1, \dots, \Sigma_m\}$  is a language  $L$  such that  $[\omega] \subseteq L$ ,  $\forall \omega \in L$ .

## III. PROBLEM FORMULATION AND APPROACH

In this section we introduce the multi-robot path planning problem with temporal constraints, and we motivate the need for solutions that are robust to uncertain robot speeds.

### A. Environment Model and Initial Formulation

Let

$$\mathcal{E} = (V, \rightarrow_{\mathcal{E}}) \quad (1)$$

be a graph, where  $V$  is the set of vertices and  $\rightarrow_{\mathcal{E}} \subseteq V \times V$  is the set of edges. In this paper,  $\mathcal{E}$  is the quotient graph of a partitioned environment, where  $V$  is a set of labels for the regions in the partition and  $\rightarrow_{\mathcal{E}}$  is the corresponding adjacency relation. For example,  $V$  can be a set of labels for the roads, intersections, and buildings in an urban-like environment and  $\rightarrow_{\mathcal{E}}$  gives their connections (see Fig. 4).

Consider a team of  $m$  robots moving in an environment modeled by  $\mathcal{E}$ . The motion capabilities of robot  $i \in \{1, \dots, m\}$  are represented by a TS  $\mathbf{T}_i = (\mathcal{Q}_i, q_i^0, \delta_i, \Pi_i, \mathcal{L}_i, w_i)$ , where  $\mathcal{Q}_i \subseteq V$ ;  $q_i^0$  is the initial vertex of robot  $i$ ;  $\delta_i \subseteq \rightarrow_{\mathcal{E}}$  is a relation modeling the capability of robot  $i$  to move among the vertices;  $\Pi_i$  is the subset of propositions  $\Pi$  assigned to the environment that can be satisfied by robot  $i$  such that  $\{\Pi_1, \dots, \Pi_m\}$  is a distribution of  $\Pi$ ;  $\mathcal{L}_i$  is a mapping from  $\mathcal{Q}_i$  to  $2^{\Pi_i}$  showing how the propositions are satisfied at vertices;  $w_i(q, q')$  captures the time for robot  $i$  to go from vertex  $q$  to  $q'$ , which we assume to be an integer. In this robotic model, robot  $i$  travels along the edges of  $\mathbf{T}_i$ , and spends zero time on the vertices. We assume that the robots are equipped with motion primitives which allow them to move from  $q$  to  $q'$  for each  $(q, q') \in \delta_i$ .

In our previous work [9] we considered the case where there is an atomic proposition  $\pi \in \Pi$ , called the *optimizing proposition*, and a multi-robot task specified by an LTL formula of the form

$$\phi := \varphi \wedge \mathbf{GF}\pi, \quad (2)$$

where  $\varphi$  can be any LTL formula over  $\Pi$ , and  $\mathbf{GF}\pi$  specifies that proposition  $\pi$  must be satisfied infinitely often. As an example, in a persistent data gathering task,  $\pi$  may be assigned to regions where data is uploaded, i.e.,  $\pi = \text{Upload}$ , while  $\varphi$  can be used to specify rules (such as traffic rules) that must be obeyed at all times during the task [8].

Our goal in [9] was to plan multi-robot paths that satisfy  $\phi$  and minimize the maximum time between satisfying instances of  $\pi$ . In data gathering, this corresponds to minimizing the maximum time between data uploads. To state this problem formally, we assume that each run  $r_i = q_i^0 q_i^1 \dots$  of  $\mathbf{T}_i$  (robot  $i$ ) starts at  $t = 0$  and generates a word  $\omega_i = \omega_i^0 \omega_i^1 \dots$  and a corresponding sequence of time instances  $\mathbb{T}_i := t_i^0 t_i^1 \dots$  such that the  $k^{\text{th}}$  symbol  $\omega_i^k = \mathcal{L}_i(q_i^k)$  is satisfied at time  $t_i^k$ . Note that, as robots spend zero time on the vertices, each  $\omega_i^k$  has a unique  $t_i^k$  which is the instant when robot  $i$  visits the corresponding vertex. To define the behavior of the team as a whole, we consider the sequences  $\mathbb{T}_i$  as sets and take the union  $\bigcup_{i=1}^m \mathbb{T}_i$  and order this set in ascending order to obtain  $\mathbb{T} := t^0 t^1, \dots$ . Then, we define  $\omega_{\text{team}} = \omega_{\text{team}}^0 \omega_{\text{team}}^1 \dots$  to be the word generated by the team of robots where the  $k^{\text{th}}$  symbol  $\omega_{\text{team}}^k$  is the union of all propositions satisfied at time  $t^k$ . Finally, we define the infinite sequence  $\mathbb{T}^\pi = \mathbb{T}^\pi(1), \mathbb{T}^\pi(2), \dots$  where  $\mathbb{T}^\pi(k)$  stands for the time instance when the optimizing proposition

$\pi$  is satisfied for the  $k^{\text{th}}$  time by the team. Thus, the problem is that of synthesizing individual optimal runs for a team of robots so that  $\omega_{\text{team}}$  satisfies  $\phi$  and  $\mathbb{T}^\pi$  minimizes

$$J(\mathbb{T}^\pi) = \limsup_{k \rightarrow +\infty} (\mathbb{T}^\pi(k+1) - \mathbb{T}^\pi(k)). \quad (3)$$

Since we consider LTL formulas containing  $\mathbf{GF}\pi$ , this optimization problem is always well-posed.

### B. Robustness and Optimality in the Field

In this paper, we are interested in the implementability of our previous approach in the case where our model is not exact in the weights of transitions. Particularly, we consider the case where the actual value of  $w_i(q, q')$  that is observed during deployment, denoted by  $\tilde{w}_i(q, q')$ , is a non-deterministic quantity that lies in the interval  $[(1 - \rho_i)w_i(q, q'), (1 + \rho_i)w_i(q, q')]$  where  $\rho_i$  is the *deviation value* of robot  $i$  which is assumed to be known a priori. In the following, we use the expression “*in the field*” to refer to the model with uncertain traveling times, and use  $x$  and  $\tilde{x}$  to denote the planned and actual values of some variable  $x$ .

The question becomes, if we use the runs generated from our previous approach in the field, will the formula  $\phi$  still be satisfied?

Given the word  $\omega_{\text{team}}$  that characterizes the planned run of the robotic team and the distribution  $\{\Pi_1, \dots, \Pi_m\}$ , the *actual* word  $\tilde{\omega}_{\text{team}}$  generated by the robotic team during its infinite asynchronous run in the field will be one of the trace equivalents of  $\omega_{\text{team}}$ , i.e.,  $\tilde{\omega}_{\text{team}} \in [\omega_{\text{team}}]$  due to the uncertainties in the traveling times of the robots. This leads to the definition of critical words.

**Definition III.1 (Critical Words).** *Given the language  $L_B$  of the Büchi automaton that corresponds to the LTL formula  $\phi$  over  $\Pi$ , and given a distribution of  $\Pi$ , we define the word  $\omega$  over  $\Pi$  to be a critical word if  $\exists \tilde{\omega} \in [\omega]$  such that  $\tilde{\omega} \notin L_B$ .*

Thus, we see that if the planned word is critical, then we may not satisfy the specification in the field. This can be formalized by noting that the optimal runs that satisfy (2) are always in a prefix-suffix form [16], where the suffix cycle is repeated infinitely often. Using this observation and Def. III.1 we can formally define the words that can violate the LTL formula during the deployment of the robotic team.

**Proposition III.2.** *If the suffix cycle of the word  $\omega_{\text{team}}$  is a critical word, then the correctness of the motion of the robotic team during its deployment cannot be guaranteed.*

*Proof.* We denote the actual word generated by the robotic team in the field by  $\tilde{\omega}_{\text{team}}$  whereas  $\omega_{\text{team}}$  stands for the planned word. Suppose that for each robot  $\rho_i = \epsilon$ , and in the suffix cycle we have  $\omega_{\text{team}}^k$  and  $\omega_{\text{team}}^{(k+\tau)}$  generated by robots  $i$  and  $j$  at positions  $k$  and  $k + \tau$  that must not be swapped, because if they do  $\tilde{\omega}_{\text{team}}$  violates  $\phi$ . Note that we are guaranteed to find such symbols as we assume the suffix cycle to be a critical word. In the worst-case, for the symbols to swap, we must have  $(1 + \epsilon)t^k > (1 - \epsilon)t^{k+\tau}$ . Solving for  $\epsilon$ , we get  $\epsilon > (t^{k+\tau} - t^k)/(t^k + t^{k+\tau})$ . However, as the suffix

is an infinite repetition of the suffix cycle,  $\lim_{k \rightarrow \infty} (t^{k+\tau} - t^k) / (t^k + t^{k+\tau}) = 0$  and  $\phi$  is violated for any  $\epsilon > 0$ . ■

In addition, we can consider the performance of the team during deployment in terms of the value of the cost function (3) observed in the field. Using the same arguments presented in Prop. III.2 it can be easily show that, the worst-case field value of (3) will be the minimum of  $(\tilde{J}_1, \dots, \tilde{J}_m)$  where  $\tilde{J}_i$  is the maximum duration between any two successive satisfactions of  $\pi$  by robot  $i$  in the field. This effectively means that there is no benefit in executing the task with multiple robots, as at some point in the future the overall performance of the team will be limited by that of a single member.

### C. Robust Problem Formulation

To characterize the field performance of the robotic team and to limit the deviation from the optimal run during deployment, we propose to use a synchronization protocol where robots can synchronize with each other only when they are at the vertices of the environment. We assume that there is an atomic proposition  $\text{Sync} \in \Pi$ , called the *synchronizing proposition*, and we consider multi-robot tasks specified using LTL formulas of the form

$$\phi_{\text{sync}} := \varphi \wedge \mathbf{GF}\pi \wedge \mathbf{GFSync}, \quad (4)$$

where  $\varphi$  can be any LTL formula over  $\Pi$ ,  $\pi$  is the optimizing proposition and  $\text{Sync}$  is the special synchronizing proposition that is satisfied only when all members of the robotic team occupy vertices at the same time. We can now formulate the problem.

**Problem III.3.** *Given a team of  $m$  robots modeled as transition systems  $\mathbf{T}_i$ ,  $i = 1, \dots, m$ , and an LTL formula  $\phi_{\text{sync}}$  over  $\Pi$  in the form (4), synthesize individual runs  $r_i$  for each robot such that  $\mathbb{T}^\pi$  minimizes the cost function (3), and  $\tilde{\omega}_{\text{team}}$ , i.e., the word observed in the field, satisfies  $\phi_{\text{sync}}$ .*

Note that the runs produced by a solution to Prob. III.3 are guaranteed not to violate  $\phi_{\text{sync}}$  even if there is a mismatch between the weights  $w_i(q, q')$  used for the solution of the problem and the actual traveling times observed in the field. Since  $\tilde{\omega}_{\text{team}}$  observed in the field is likely to be sub-optimal, we will also seek to bound the deviation from optimality in the field.

### D. Solution Outline

In [9], we showed that the joint behavior of a robotic team can be captured by a region automaton. A region automaton, as defined next, is a finite dimensional transition system that captures the relative positions of the members of the robotic team. This information is then used for computing optimal trajectories.

**Definition III.4 (Region Automaton).** *The region automaton  $\mathbf{R}$  is a TS (Def. II.1)  $\mathbf{R} := (\mathcal{Q}_R, q_R^0, \delta_R, \Pi_R, \mathcal{L}_R, w_R)$ , where*

- (i)  $\mathcal{Q}_R$  is the set of states of the form  $(q, r)$  such that

- (a)  $q$  is a tuple of state pairs  $(q_1 q'_1, \dots, q_m q'_m)$  where the  $i^{\text{th}}$  element  $q_i q'_i$  is a source-target state pair from  $\mathcal{Q}_i$  of  $\mathbf{T}_i$  meaning robot  $i$  is currently on its way from  $q_i$  to  $q'_i$ , and
- (b)  $r$  is a tuple of clock values  $(x_1, \dots, x_m)$  where the  $i^{\text{th}}$  element denotes the time elapsed since robot  $i$  left state  $q_i$ .
- (ii)  $q_R^0$  is the initial state that has zero-weight transitions to all those states in  $\mathcal{Q}_R$  with  $r = (0, \dots, 0)$  and  $q = (q_1 q'_1, \dots, q_m q'_m)$  such that  $q_i^0$  is the initial state of  $\mathbf{T}_i$  and  $(q_i^0, q'_i) \in \delta_i$ .
- (iii)  $\delta_R$  is the transition relation such that a transition from  $(q, r)$  to  $(q', r')$  exists if and only if
  - (a)  $(q_i, q'_i), (q'_i, q''_i) \in \delta_i$  for all changed state pairs where the  $i^{\text{th}}$  element  $q_i q'_i$  in  $q$  changes to  $q'_i q''_i$  in  $q'$ ,
  - (b)  $w_i(q_i, q'_i) - x_i$  of all changed state pairs are equal to each other and are strictly smaller than those of unchanged state pairs, and
  - (c) for all changed state pairs corresponding  $x'_i$  in  $r'$  becomes  $x'_i = 0$  and all other clock values in  $r$  are incremented by  $w_i(q_i, q'_i) - x_i$  in  $r'$ .
- (iv)  $\Pi_R = \cup_{i=1}^m \Pi_i$  is the set of propositions;
- (v)  $\mathcal{L}_R : \mathcal{Q}_R \rightarrow 2^{\Pi_R}$  is a map giving the set of atomic propositions satisfied in a state. For a state with  $q = (q_1 q'_1, \dots, q_m q'_m)$ ,  $\mathcal{L}_R((q, r)) = \cup_{i=1}^m \mathcal{L}_i(q_i)$ ;
- (vi)  $w_R : \delta_R \rightarrow \mathbb{R}_{\geq 0}$  is a map that assigns a non-negative weight to each transition such that  $w_R((q, r), (q', r')) = w_i(q_i, q'_i) - x_i$  for each state pair that has changed from  $q_i q'_i$  to  $q'_i q''_i$  with a corresponding clock value of  $x'_i = 0$  in  $r'$ .

**Example III.5.** Fig. 2 illustrates the region automaton  $\mathbf{R}$  that corresponds to the robots modeled with  $\mathbf{T}_1$  and  $\mathbf{T}_2$  given in Fig. 1. There is a transition from  $((ba, bc), (0, 0))$  to  $((ba, cb), (1, 0))$  with weight 1 in  $\mathbf{R}$  because  $(b, c) \in \delta_2$ ,  $w_2(b, c) = 1$ , and  $w_1(b, a) \neq 1$ .

Our solution to Problem III.3 can be outlined as follows:

- (i) We check if the LTL formula  $\phi_{\text{sync}}$  is trace-closed guaranteeing that it will not be violated in the field (See Sec. IV-A);
- (ii) We prepare the serialized region automaton of the robotic team with synchronization points by modifying the output of our earlier algorithm OBTAIN-REGION-AUTOMATON [9] (See Sec. IV-B);
- (iii) We find optimal runs on individual  $\mathbf{T}_i$ s using the OPTIMAL-RUN algorithm we previously developed in [16] and use a synchronization protocol to calculate an upper bound on the cost function (3) for given deviation values to obtain the solution to Prob. III.3 (See Sec. IV-C).

## IV. PROBLEM SOLUTION

In this section, we explain each step of the solution to Prob. III.3 in detail. In the following, we use a simple example to illustrate ideas as we develop the theory for the

general case. We present an experimental evaluation of our approach considering a more realistic scenario in Sec. V.

#### A. Trace-Closedness of the Original Formula

Prop. IV.1 shows how trace-closedness of  $\phi_{sync}$  guarantees correctness in the field. In the following, we say an LTL formula  $\phi_{sync}$  is trace-closed if the language  $L_B$  of the corresponding Büchi automaton is trace-closed in the sense of Def. II.8.

**Proposition IV.1.** *If the general specification  $\phi_{sync}$  is a trace-closed formula with respect to the distribution given by the robots' capabilities, then it will not be violated in the field due to uncertainties in the speeds of the robots.*

*Proof.* From Defs. II.8 and III.1, we know that if we can find a run that satisfies a trace-closed LTL formula, then the word  $\omega_{team}$  produced by the run will not be a critical word. Since  $\omega_{team}$  is not a critical word,  $\nexists \tilde{\omega}_{team} \in [\omega_{team}]$  such that  $\tilde{\omega}_{team} \notin L_B$ . Thus, regardless of the  $\rho_i$  values of the robots,  $\phi$  will not be violated in the field due to robot timing errors as any  $\tilde{\omega}_{team} \in [\omega_{team}]$  will also be in  $L_B$ . ■

Thus, in order to guarantee correctness in the field, we first check that  $\phi_{sync}$  is trace-closed using an algorithm adapted from [17]. However, as trace-closedness is not well-defined for words over  $2^\Pi$ , we construct a Büchi automaton whose language  $L_B$  is over the set  $\Pi$ .

**Example IV.2.** Fig. 1 illustrates the environment where two robots are expected to satisfy a task given by a formula in the form of (4) where  $\varphi = \mathbf{GFr1P} \wedge \mathbf{GFr2P}$ ,  $\Pi_1 = \{r1P, \pi, \text{Sync}\}$ ,  $\Pi_2 = \{r2P, \pi, \text{Sync}\}$ , and  $\Pi = \{r1P, r2P, \pi, \text{Sync}\}$ .

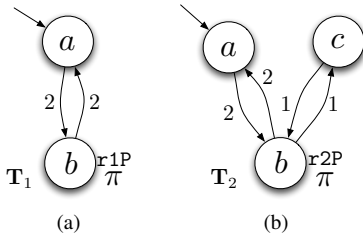


Fig. 1: TS's  $T_1$  and  $T_2$  of two robots in an environment with three vertices. The states correspond to vertices  $\{a, b, c\}$ , the edges represent the motion capabilities of each robot, and the weights represent the traveling times between any two vertices. The propositions  $r1P$ ,  $r2P$  and  $\pi$  are shown next to the vertices where they can be satisfied by the robots.

After checking that  $\phi_{sync}$  is trace-closed, we proceed by obtaining the serialized region automaton with synchronization points where the Sync proposition is satisfied.

#### B. Obtaining the Serialized Region Automaton with Synchronization Points

If  $\phi_{sync}$  is a trace-closed formula, we obtain the region automaton that captures the joint behavior of the robotic team using OBTAIN-REGION-AUTOMATON [9]. Next, using Alg. 1, we first introduce synchronization states by adding

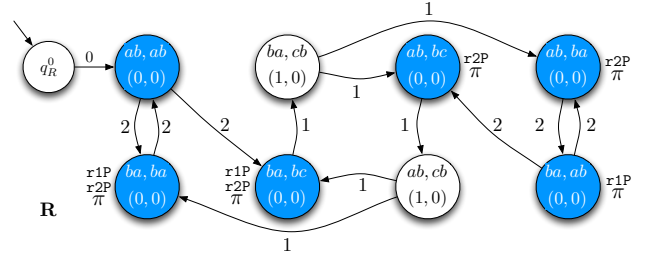


Fig. 2: Region automaton obtained using OBTAIN-REGION-AUTOMATON [9] that captures the joint behavior of the robotic team given in Fig. 1. Sync states where all robots occupy vertices, i.e., states with all zero clock values, are highlighted in blue.

the special Sync proposition to the states where all robots occupy some vertex in their TS's simultaneously, i.e., states with  $r = (0, \dots, 0)$ . Note that, these are the states that will be used to calculate a bound on optimality when the robots are deployed in the field. We then expand the states where multiple propositions are satisfied simultaneously to obtain  $R_{ser}$  where at most one proposition is satisfied at each state. This ensures that languages of both the Büchi automaton that corresponds to  $\phi_{sync}$  and  $R_{ser}$  are over  $\Pi$ .

**Example IV.2 Revisited.** Fig. 2 illustrates the region automaton  $R$  that captures the joint behavior of the team given in Fig. 1. The serialized region automaton with synchronization points  $R_{ser}$  that corresponds to  $R$  is given in Fig. 3.

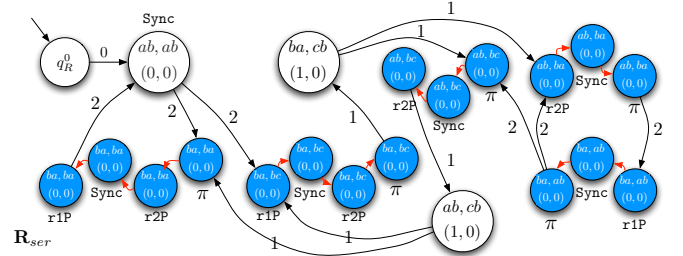


Fig. 3: Serialized region automaton with synchronization states obtained after applying Alg. 1 to  $R$  in Fig. 2. New states introduced after serialization are highlighted in blue. Red arrows stand for zero-weight transitions.

**Remark IV.3.** Since  $\phi_{sync}$  is trace-closed, the serialization can be done in any order. Since all possible orderings belong to the same trace-equivalent class, they do not affect the satisfaction of the formula.

#### C. Finding the Robust Optimal Run and the Optimality Bound

After obtaining the serialized region automaton  $R_{ser}$ , we find an optimal run  $r_R^*$  on  $R_{ser}$  that minimizes the cost function (3) using our earlier OPTIMAL-RUN algorithm [16]. The optimal run  $r_R^*$  is always in a prefix-suffix form (Def. II.4). Furthermore, as  $r_R^*$  satisfies  $\phi_{sync}$ , it has at least one synchronization point in its suffix cycle, which we assume to start with a synchronization point.

---

**Algorithm 1: SERIALIZE-REGION-AUTOMATON**

---

**Input:** A region automaton  $\mathbf{R}$  obtained using OBTAIN-REGION-AUTOMATON .

**Output:**  $\mathbf{R}_{ser}$ , the serialized region automaton with synchronization states, such that at most one proposition is satisfied at each state.

```

1 foreach State  $\{q, r\}$  in  $\mathbf{R}$  do
2   if  $r = (0, \dots, 0)$  then
3     Add Sync to propositions satisfied in  $\{q, r\}$ .
4    $k \leftarrow$  Number of propositions satisfied in  $\{q, r\}$ .
5   if  $k > 1$  then
6      $propsTuple \leftarrow$  The tuple  $(p_1, \dots, p_k)$  of
       propositions satisfied in  $\{q, r\}$ .
7     Copy  $\{q, r\}$   $k$  times to obtain  $\{q, r\}'_1, \dots, \{q, r\}'_k$ .
8     foreach  $i = 1, \dots, k$  do
9        $\mathcal{L}(\{q, r\}'_i) \leftarrow propsTuple[i]$ .
10      if  $i < k$  then
11        Add  $\{q, r\}'_i \rightarrow \{q, r\}'_{i+1}$  to  $\delta_R$  with zero
          weight.
12    Re-direct all incoming transitions of  $\{q, r\}$  to
       $\{q, r\}'_1$ .
13    Originate all outgoing transitions of  $\{q, r\}$  from
       $\{q, r\}'_k$ .
14    Remove  $\{q, r\}$  from  $\mathcal{Q}_R$ .
```

---

**Definition IV.4 (Projection of a run on  $\mathbf{R}$  to  $\mathbf{T}_i$ s).** Given  $\mathbb{T}$  and the corresponding run  $r_R$  on  $\mathbf{R}_{ser}$  where

$$r_R = ((q_1^0 q_1^1, \dots, q_m^0 q_m^1), (x_1^0, \dots, x_m^0)) \\ ((q_1^1 q_1^2, \dots, q_m^1 q_m^2), (x_1^1, \dots, x_m^1)) \dots,$$

we define its projection on  $\mathbf{T}_i$  as run  $r_i = q_i^0 q_i^1 \dots$  for all  $i = 1, \dots, m$ , where  $q_i^k$  only appears in  $r_i$  if  $x_i^k = 0$  and  $\mathbb{T}(k) \neq \mathbb{T}(k+1)$ .

In [9] we show that the individual runs  $r_i$  obtained by the projection in Def. IV.4 are equivalent to the region automaton run  $r_R$  in the sense that they produce the same word  $\omega_{team}$ . Using Def. IV.4, we project the optimal run  $r_R^*$  to individual  $\mathbf{T}_i$ s to obtain the set of optimal individual runs  $\{r_1^*, \dots, r_m^*\}$ . As the robots execute their infinite runs in the field, they synchronize with each other at the synchronization point following the protocol given in Alg. 2 ensuring that they start each new suffix cycle in a synchronized way. Using this protocol, we can define a bound on optimality, i.e., the value of the cost function (3) observed in the field, as given in the following proposition.

**Proposition IV.5.** Suppose that each robot's deviation value is bounded by  $\rho > 0$  (i.e.,  $\rho_i \leq \rho$  for all robots  $i$ ), and let  $J(\mathbb{T}^\pi)$  be the cost of the planned robot paths. Then, if the robots follow the protocol given in Alg. 2 the field value of the cost satisfies

$$\overline{J}(\mathbb{T}^\pi) \leq J(\mathbb{T}^\pi) + \rho(J(\mathbb{T}^\pi) + 2d_s),$$

---

**Algorithm 2: SYNC-RUN**

---

**Input:** A run  $r_k$  of robot  $k$  in the prefix-suffix form with at least one synchronization point in its suffix cycle.

```

1 begin
2    $syncPoint \leftarrow$  First synchronization point in the
     suffix.
3    $teamFlags \leftarrow (0, \dots, 0)$ .
4   while True do
5     if  $syncMessage$  received from robot  $i$  then
6        $teamFlags[i] \leftarrow 1$ .
7     if  $currentState = syncPoint$  then
8       Stop
9       Broadcast  $syncMessage$ .
10       $teamFlags[k] \leftarrow 1$ .
11     if  $teamFlags = (1, \dots, 1)$  then
12        $teamFlags \leftarrow (0, \dots, 0)$ .
13     Continue executing  $r_k$ .
```

---

where  $d_s$  is the planned duration of the suffix cycle.

*Proof.* In the following, we take the suffix to begin at a synchronization point. The suffix consists of an infinite number of repetitions of the suffix cycle, which we denote  $S_c$ . Let  $d_s$  be the planned duration of  $S_c$ , let  $n_s$  be the number of optimizing propositions satisfied in  $S_c$ . Let us redefine  $t = 0$  to be the time when the suffix starts, and let  $\mathbb{T}^\pi$  be a sequence of length  $n_s$  recording the  $n_s$  times that the optimizing proposition is satisfied on the first repetition of  $S_c$ . Note that, as we consider infinite runs and as the process restarts itself at the beginning of each  $S_c$  by means of the synchronization protocol given in Alg. 2, we only need to consider the first repetition of  $S_c$ . We first define

$$\underline{T}^i = \mathbb{T}^\pi(i)(1 - \rho) \\ \overline{T}^i = \mathbb{T}^\pi(i)(1 + \rho) \\ t^w = d_s(1 + \rho)$$

where,  $\underline{T}^i$  and  $\overline{T}^i$  are the earliest and latest times that the  $i$ th optimizing proposition can be satisfied, respectively. The value  $t^w$  is the latest time that the second repetition of  $S_c$  can begin. Then, for  $0 < i \leq n_s$ , the worst-case time between satisfying the  $i$ th optimizing proposition and the  $(i+1)$ th optimizing proposition is

$$\tau^{i,i+1} = \begin{cases} \overline{T}^{i+1} - \underline{T}^i & \text{if } 0 < i < n_s, \\ t^w + \overline{T}^1 - \underline{T}^{n_s} & \text{if } i = n_s. \end{cases} \quad (5)$$

Next, in the planned paths, multiple robots may simultaneously satisfy the  $i$ th optimizing proposition. In the field, these satisfactions will not occur simultaneously. The maximum amount of time between the first and last of these satisfying instances for the  $i$ th proposition, for  $0 < i \leq n_s$ , is

$$\tau^i = \overline{T}^i - \underline{T}^i. \quad (6)$$

Finally, using (5) and (6) we obtain the upper bound on the value of the cost function (3) that will be observed during deployment as

$$\overline{J(\mathbb{T}^\pi)} = \max\{\max_i\{\tau^{i,i+1}\}, \max_i\{\tau^i\}\}. \quad (7)$$

Substituting the definitions for  $\overline{T^i}$ ,  $\underline{T^i}$ , and  $t^w$  into (5) we obtain  $\tau^{i,i+1} =$

$$\begin{cases} \overline{T}^\pi(i+1)(1+\rho) - \overline{T}^\pi(i)(1-\rho) & \text{if } 0 < i < n_s, \\ (1+\rho)(d_s + \overline{T}^\pi(1)) - \overline{T}^\pi(n_s)(1-\rho) & \text{if } i = n_s \end{cases}$$

But, we have that  $J(\mathbb{T}^\pi) \geq \overline{T}^\pi(i+1) - \overline{T}^\pi(i)$ , and  $J(\mathbb{T}^\pi) \geq d_s + \overline{T}^\pi(1) - \overline{T}^\pi(n_s)$ . In addition,  $\overline{T}^\pi(1) \leq J(\mathbb{T}^\pi)$  and  $\overline{T}^\pi(i) \leq d_s$  for all  $i \in \{2, \dots, n_s\}$ . Using these expressions we obtain

$$\tau^{i,i+1} \leq J(\mathbb{T}^\pi) + \rho(J(\mathbb{T}^\pi) + 2d_s).$$

Similarly, we get

$$\tau^i \leq J(\mathbb{T}^\pi) + 2\rho d_s,$$

and thus  $\overline{J(\mathbb{T}^\pi)} \leq J(\mathbb{T}^\pi) + \rho(J(\mathbb{T}^\pi) + 2d_s)$ . ■

**Remark IV.6.** In Prop. IV.5, we have provided a conservative bound for ease of presentation. However, we can calculate an exact bound on the field value of the cost  $J(\mathbb{T}^\pi)$  using a treatment similar to the proof of Prop IV.5.

**Example IV.2 Revisited.** For the example we have shown throughout this section, applying Alg. OPTIMAL-RUN [16] to  $\mathbf{R}_{ser}$  given in Fig. 2 and the formula  $\phi_{sync} := \mathbf{GF}r1P \wedge \mathbf{GF}r2P \wedge \mathbf{GF}\pi \wedge \mathbf{GF}Sync$  results in the optimal run with the prefix

| $\mathbb{T}$           | 0              | 2              | 2              | 2              | 2              | 3              |
|------------------------|----------------|----------------|----------------|----------------|----------------|----------------|
| $r_R^*$                | ab,ab<br>(0,0) | ba,bc<br>(0,0) | ba,bc<br>(0,0) | ba,bc<br>(0,0) | ba,bc<br>(0,0) | ba,cb<br>(1,0) |
| $\mathcal{L}_R(\cdot)$ | Sync           | r1P            | Sync           | r2P            | $\pi$          |                |

and the suffix cycle

| $\mathbb{T}$           | 4              | 4              | 4              | 6              | 6              | 6              |
|------------------------|----------------|----------------|----------------|----------------|----------------|----------------|
| $r_R^*$                | ab,ba<br>(0,0) | ab,ba<br>(0,0) | ab,ba<br>(0,0) | ba,ab<br>(0,0) | ba,ab<br>(0,0) | ba,ab<br>(0,0) |
| $\mathcal{L}_R(\cdot)$ | r2P            | Sync           | $\pi$          | r1P            | Sync           | $\pi$          |

which will be repeated an infinite number of times. In the table above, the rows correspond to the times when transitions occur, the run  $r_R^*$ , and the satisfying atomic propositions, respectively. For this example,  $\mathbb{T}^\pi = 2, 4, 6, 8, 10, \dots$  and the cost as defined in (3) is  $J(\mathbb{T}^\pi) = 2$ . Furthermore, when the robotic team is deployed in the field, this cost is bounded from above by 2.5 for  $\rho_1 = \rho_2 = 0.05$  as given by Prop. IV.5.

Applying Def. IV.4 to  $r_R^*$  we have the following individual runs:

| $\mathbb{T}$ | 0 | 2 | 3 | 4 | 6 | 8 | 10 | ... |
|--------------|---|---|---|---|---|---|----|-----|
| $r_1^*$      | a | b |   | a | b | a | b  | ... |
| $r_2^*$      | a | b | c | b | a | b | a  | ... |

Note that, at time  $t = 3$ , the second robot has arrived at  $c$  while the first robot is still traveling from  $b$  to  $a$ , therefore the clock of the first robot is not zero at this time, i.e.,  $x_1 \neq 0$ , and  $b$  does not appear in  $r_1^*$  at time  $t = 3$ .

We finally summarize our approach in Alg. 3, show that this algorithm indeed gives a solution to Prob. III.3 and analyze the overall complexity of our approach.

---

**Algorithm 3: ROBUST-MULTI-ROBOT-OPTIMAL-RUN**

---

**Input:**  $m$   $\mathbf{T}_i$ 's and a global LTL specification  $\phi_{sync}$  of form (4).

**Output:** A set of robust optimal runs  $\{r_1^*, \dots, r_m^*\}$  that satisfies  $\phi_{sync}$ , minimizes (3), and the bound on the performance of the team in the field.

```

1 begin
2    $\phi_{sync} := \varphi \wedge \mathbf{GF}\pi \wedge \mathbf{GF}Sync$ .
3   if  $\phi_{sync}$  is trace-closed then
4     Obtain the region automaton  $\mathbf{R}$  using
       OBTAIN-REGION-AUTOMATON [9].
5     Obtain  $\mathbf{R}_{ser}$  using
       SERIALIZE-REGION-AUTOMATON .
6     Find the optimal run  $r_R^*$  applying OPTIMAL-RUN
       [16] to  $\mathbf{R}_{ser}$  and  $\phi_{sync}$ .
7     Obtain individual runs from  $r_R^*$  using Def. IV.4.
8     Find the bound on optimality as given in
       Prop. IV.5.
9   else
10    Abort.
```

---

**Proposition IV.7.** Alg. 3 solves Prob. III.3.

*Proof.* Note that Alg. 3 combines all steps outlined in this section. The planned word  $\omega_{team}$  generated by the entire team satisfies  $\phi_{sync}$  and minimizes (3), as shown in [9]. Furthermore, since  $\phi_{sync}$  is trace-closed, the optimal satisfying run is guaranteed not to violate  $\phi_{sync}$  in the field due to timing errors as given in Prop. IV.1. Therefore,  $\{r_1^*, \dots, r_m^*\}$  as obtained from Alg. 3 is the solution to Prob. III.3. ■

**Proposition IV.8.** For the case where a group of  $m$  identical robots are expected to satisfy an LTL specification  $\phi$  in a common environment with  $\Delta$  edges and a largest edge weight of  $W$ , the worst-case complexity of Alg. 3 is  $O((\Delta \cdot W)^{3m} \cdot 2^{O(|\phi|)})$ .

*Proof.* From [9], the number of states of the region automaton  $\mathbf{R}$  is bounded by

$$\left( \prod_{i=1}^m |\delta_i| \right) \left( \prod_{i=1}^m W_i - \prod_{i=1}^m (W_i - 1) \right) + 1$$

where  $m$  is number of robots and  $W_i$  is largest edge weight in TS  $\mathbf{T}_i$  of robot  $i$ . Then, for the above mentioned case, the worst-case size of the region automaton is  $O((\Delta \cdot W)^m)$ . In [8], the authors give the worst-case complexity of the OPTIMAL-RUN algorithm as  $O(|T|^3 \cdot 2^{O(|\phi|)})$  where  $|T|$  is the number of states of the input transition system and  $|\phi|$  is the length of the LTL specification. Therefore, the worst-case complexity of Alg. 3 becomes  $O((\Delta \cdot W)^{3m} \cdot 2^{O(|\phi|)})$ . ■



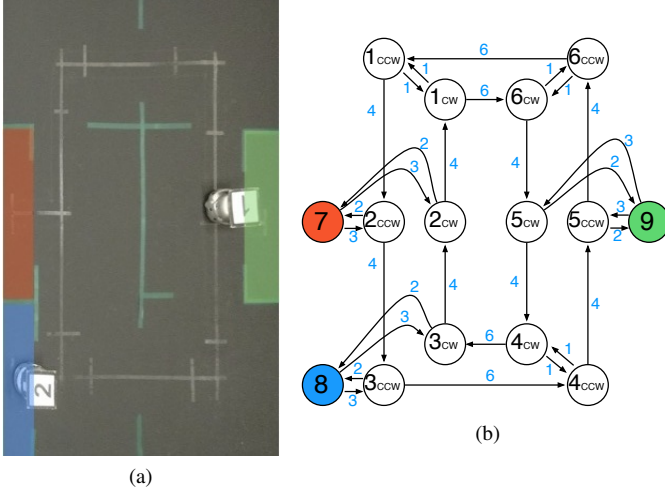


Fig. 4: (a) Road network used in the experiments (b) The model of the road network with weights shown in blue. 1 time unit in this model corresponds to 3 seconds. The red and blue regions are data gathering locations of robots 1 and 2, respectively and the green region is the common upload location. CW and CCW stand for clockwise and counter-clockwise, respectively.

## V. IMPLEMENTATION AND CASE STUDIES

We implemented Alg. 3 in objective-C as the software package LTL ROBUST OPTIMAL MULTI-ROBOT PLANNER (LROMP) and used it in conjunction with our earlier OPTIMAL-RUN [16] algorithm to obtain robust and optimal trajectories for robots performing persistent data gathering missions in a road network environment. The software package, available at <http://hyness.bu.edu/Software.html>, utilizes the dot tool [18] to visualize transition systems and the OPTIMAL-RUN algorithm uses the LTL2BA software [19] to convert LTL specifications to Büchi automata. Following the steps detailed in Sec. IV, the software first creates the serialized region automaton with synchronization states  $\mathbf{R}_{ser}$  using  $\mathbf{T}_i$ s defined by the user and exports an M-file which defines  $\mathbf{R}_{ser}$  in Matlab. Next,  $\phi_{sync}$  is checked for trace-closedness, after which OPTIMAL-RUN algorithm is executed in Matlab to find the optimal run  $r_R^*$  on  $\mathbf{R}_{ser}$ . Finally, an upper bound on the field value of the cost function (3) is computed and  $r_R^*$  is projected to individual  $\mathbf{T}_i$ ,  $i = 1, \dots, m$ , to obtain the solution to Prob. III.3.

Fig. 4 illustrates our experimental platform, which is a road network consisting of roads, intersections, and task locations. The figure also shows the transition system that models the motion of the robots on this road network where 1 time unit corresponds to 3 seconds. In the following, the transition systems  $\mathbf{T}_i$  are identical except for their initial states and the sets of propositions that can be satisfied at states.

In our experiments, we consider a persistent monitoring task where two robots with deviation values of  $\rho_1 = 0.09$ ,  $\rho_2 = 0.04$  repeatedly gather and upload data and the maximum time in between any two data uploads must be minimized. We require robots 1 and 2 to gather data at 7 and 8 in

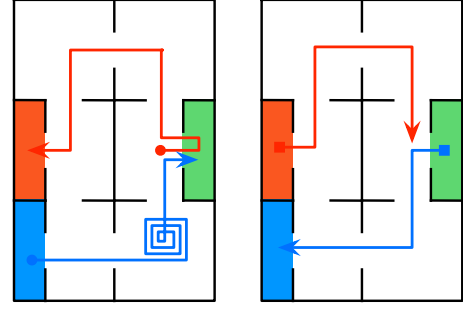


Fig. 5: Team trajectories used in the experiments. The red and blue regions are data gathering locations of robots 1 and 2, respectively and the green region is the common upload location. The circles on the left show the sync point, i.e., the beginning of the suffix cycle, on the trajectories of the robots.

Fig. 4, respectively and upload the data at 9. We define  $\Pi = \{\mathbf{R1Gather}, \mathbf{R1Upload}, \mathbf{R2Gather}, \mathbf{R2Upload}, \mathbf{Upload}, \mathbf{Sync}\}$  and assign the atomic propositions as

$$\begin{aligned} \mathcal{L}_1(7) &= \{\mathbf{R1Gather}\}, \mathcal{L}_1(9) = \{\mathbf{R1Upload}, \mathbf{Upload}\} \\ \mathcal{L}_2(8) &= \{\mathbf{R2Gather}\}, \mathcal{L}_2(9) = \{\mathbf{R2Upload}, \mathbf{Upload}\}. \end{aligned}$$

where Upload is set as the optimizing proposition ( $\pi$  as in formula (4)) due to the task specification. Next, we forbid data uploads unless robots have something to upload using the LTL formula

$$\begin{aligned} \varphi &= \mathbf{G}(\mathbf{R1Upload} \Rightarrow \mathbf{X}(\neg \mathbf{R1Upload} \mathcal{U} \mathbf{R1Gather})) \\ &\quad \wedge \mathbf{G}(\mathbf{R2Upload} \Rightarrow \mathbf{X}(\neg \mathbf{R2Upload} \mathcal{U} \mathbf{R2Gather})). \end{aligned}$$

Our overall LTL formula in the form of (4) is

$$\phi_{sync} = \varphi \wedge \mathbf{G} \mathbf{F} \mathbf{Upload} \wedge \mathbf{G} \mathbf{F} \mathbf{Sync}. \quad (8)$$

Running our algorithms on an iMac i5 quad-core computer, we obtain the robust optimal trajectory as illustrated in Fig. 5. The algorithm ran for 35 minutes, and the region automaton  $\mathbf{R}_{ser}$  had 5224 states. The value of the cost function was 19 time units (57 seconds) with an upper-bound of 27.55 time units (82.65 seconds), meaning that the maximum time in between data uploads would be less than 82.65 seconds in the field. This result was experimentally verified in our robotic test-bed and the maximum time in between data uploads was measured to be 64 seconds (21.3 time units) during a run of 13 minutes. In order to confirm and demonstrate the effectiveness of our approach, we executed the same trajectory without any synchronization. After approximately 6.5 minutes, the maximum time in between data uploads was measured to be 92 seconds (30.7 time units), much worse than what is provided by our approach. Our video submission accompanying the paper displays the robot trajectories for both cases.

It is interesting to note that, in the optimal solution the second robot spends extra time spinning between states  $4_{CW}$  and  $4_{CCW}$  (Figs. 4b, 5). This behavior is actually time-wise optimal as it decreases the maximum time between successive satisfying instances of the optimizing proposition.



## VI. CONCLUSIONS

In this paper we presented and experimentally evaluated a method for planning robust optimal trajectories for a team of robots that satisfy a common temporal logic mission specification. Our method is robust to uncertainties in the traveling times of each robot, and thus has practical value in applications where multiple robots must perform a series of tasks collectively in a common environment. We considered trace-closed temporal logic formulas with optimizing and synchronizing propositions that must be repeatedly satisfied. In the absence of timing errors, the motion plan delivered by our method is optimal in the sense that it minimizes the maximum time between satisfying instances of the optimizing proposition. If the traveling times observed in the field deviate from those given by the transition systems of the robots, our method guarantees that the mission specification is never violated and provides an upper bound on the ratio between the performance in the field and the optimal performance.

## ACKNOWLEDGMENTS

We thank Jennifer Marx at Boston University for her work on the experimental platform.

## REFERENCES

- [1] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 153–158.
- [2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
- [3] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Hybrid systems: Computation and Control*, Stockholm, Sweden, 2010, pp. 101–110.
- [4] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Motion planning with hybrid dynamics and temporal goals," in *IEEE Conf. on Decision and Control*, 2010, pp. 1108–1115.
- [5] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. LaValle, "Controlling wild bodies using linear temporal logic," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.
- [6] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Logic in Computer Science*, 1986, pp. 322–331.
- [7] G. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 279–295, 1997.
- [8] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," *International Journal of Robotics Research*, vol. 30, pp. 1695–1708, Dec. 2011.
- [9] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," pp. 3087–3092, Sep. 2011.
- [10] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.
- [11] R. Milner, *Communication and concurrency*. Prentice-Hall, 1989.
- [12] A. Puri, "An undecidable problem for timed automata," *Discrete Event Dynamic Systems*, vol. 9, no. 2, pp. 135–146, 2000.
- [13] —, "Dynamical properties of timed automata," *Discrete Event Dynamic Systems*, vol. 10, no. 1-2, pp. 87–113, 2000.
- [14] L. C. G. J. M. Habets and J. H. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, pp. 21–35, 2004.
- [15] Y. Chen, X. C. Ding, and C. Belta, "A formal approach to the deployment of distributed robotic teams," *IEEE Transactions on Robotics*, 2011, to appear.
- [16] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning under temporal logic constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Taipei, Taiwan, Oct. 2010, pp. 3288–3293.
- [17] D. Peled, T. Wilke, and P. Wolper, "An algorithmic approach for checking closure properties of temporal logic specifications and omega-regular languages," *Theor. Comput. Sci.*, vol. 195, no. 2, pp. 183–203, 1998.
- [18] "Graphviz - graph visualization software," <http://www.graphviz.org/>.
- [19] "LTL2BA," <http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/index.php>.